INTERPOLATIVE DECOMPOSITION BUTTERFLY FACTORIZATION

QIYUAN PANG\textsuperscript{1}, KENNETH L. HO\textsuperscript{2}, AND HAIZHAO YANG\textsuperscript{3}

Abstract. This paper introduces a “kernel-independent” interpolative decomposition butterfly factorization (IDBF) as a data-sparse approximation for matrices that satisfy a complementary low-rank property. The IDBF can be constructed in $O(N \log N)$ operations for an $N \times N$ matrix via hierarchical interpolative decompositions (IDs) if matrix entries can be sampled individually and each sample takes $O(1)$ operations. The resulting factorization is a product of $O(\log N)$ sparse matrices, each with $O(N)$ nonzero entries. Hence, it can be applied to a vector rapidly in $O(N \log N)$ operations. IDBF is a general framework for nearly optimal fast matrix-vector multiplication (matvec), which is useful in a wide range of applications, e.g., special function transformation, Fourier integral operators, and high-frequency wave computation. Numerical results are provided to demonstrate the effectiveness of the butterfly factorization and its construction algorithms.

Key words. data-sparse matrix, butterfly factorization, interpolative decomposition, operator compression, Fourier integral operators, high-frequency integral equations

AMS subject classifications. 44A55, 65R10, 65T50

DOI. 10.1137/19M1294873

1. Introduction. One of the key computational tasks in scientific computing is to rapidly evaluate dense matrix-vector multiplication (matvec). Given a dense matrix $K \in \mathbb{C}^{N \times N}$ and a vector $x \in \mathbb{C}^N$, it takes $O(N^2)$ operations to naively compute the vector $y = Kx \in \mathbb{C}^N$. There has been extensive research on constructing data-sparse representations of structured matrices (e.g., low-rank matrices [1, 2, 3, 4], $H$ matrices [5, 6, 7], $\mathcal{H}^2$ matrices [8, 9], HSS matrices [10, 11], complementary low-rank matrices [12, 13, 14, 15, 16, 17], FMM [18, 19, 20, 21, 22, 23, 24, 25], directional low-rank matrices [26, 27, 28, 29], and the combination of these matrices [30, 31]) with the goal of obtaining linear or nearly linear scaling matvec. In particular, this paper concerns nearly optimal matvec for complementary low-rank matrices.

A wide range of transforms in harmonic analysis [13, 14, 32, 33, 34, 35] and integral equations in the high-frequency regime [30, 31] admit a matrix or its submatrices satisfying a complementary low-rank property. For a 1D complementary low-rank matrix, its rows are typically indexed by a point set $X \subset \mathbb{R}$ and its columns by another point set $\Omega \subset \mathbb{R}$. Associated with $X$ and $\Omega$ are two trees $T_X$ and $T_\Omega$ constructed by dyadic partitioning of each domain. Both trees have the same level $L + 1 = O(\log N)$, with the top root being the 1st level and the bottom leaf being the $(L + 1)$th level. We say a matrix satisfies the complementary low-rank property if, for any node $A$ at level $\ell$ in $T_X$ and any node $B$ at level $L + 2 - \ell$, the submatrix $K_{A,B}$ of $K$, obtained by restricting the rows of $K$ to the points in node $A$ and the columns to the points in node $B$, is numerically low-rank; that is, given a precision $\epsilon$, there exists an approximation...
of $K'_{A,B}$ with the 2-norm of the error bounded by $\epsilon$ and the rank $k$ bounded by a polynomial in $\log N$ and $\log 1/\epsilon$.

Points in $X \times \Omega$ may be nonuniformly distributed. Hence, submatrices $\{K^\ell_{A,B}\}_{A,B}$ at the same level $\ell$ may have different sizes, but they have almost the same rank. If the point distribution is uniform, then at the $\ell$th level starting from the root of $T_X$, submatrices have the same size $\frac{N}{2^\ell} \times 2^\ell - 1$. See Figure 1 for an illustration of low-rank submatrices in a 1D complementary low-rank matrix of size $16 \times 16$ with uniform point distributions in $X$ and $\Omega$. It is easy to generalize the complementary low-rank matrices to higher dimensional space as in [16]. For simplicity, we only present interpolative decomposition butterfly factorization (IDBF) for the 1D case with uniform point distributions and leave to the reader the extension to nonuniform point distributions and higher dimensional cases.

This paper introduces IDBF as a data-sparse approximation for kernel matrices that satisfy the complementary low-rank property. IDBF can be constructed in $O(\frac{k^3}{n_0} N \log N)$ operations for an $N \times N$ matrix $K$, with a local rank parameter $k$ and a leaf size parameter $n_0$ via hierarchical linear interpolative decompositions (IDs), if each matrix entry can be sampled individually in $O(1)$ operations and the matrix itself admits good prior proxy points. The resulting factorization is a product of $O(\log N)$ sparse matrices, each of which contains $O(\frac{k^3}{n_0} N)$ nonzero entries as follows:

$$K \approx U^L U^{L-1} \cdots U^h S^h V^h \cdots V^{L-1} V^L,$$

where $h = L/2$, and the level $L$ is assumed to be even. Hence, it can be applied to a vector rapidly in $O(\frac{k^3}{n_0} N \log N)$ operations.

This paper mainly focuses on the kernel-independent butterfly factorization in the sense that the factorization does not rely on the explicit formula of the kernel matrix $K$ but assumes that the kernel matrix $K$ is the discretization of certain kernels; i.e., rows and columns are associated with points in the discretization. Most previous work in the literature requires expensive precomputation time, e.g., [13, 14, 16], which motivates the work in this paper. After the completion of this paper, the authors became aware of [36], which addresses similar questions. The algorithm in [36] was also applied in [12, 30, 31], resulting in nearly linear scaling fast matvec therein. The algorithm in [36] is organized slightly differently from our algorithm but shares many aspects with it. In our algorithm, linear scaling IDs are applied to low-rank submatrices instead of low-rank approximations based on a sampling technique in [37], the latter of which might not be as accurate and stable as IDs since it shares the same spirit of CUR decomposition.

2. Interpolative decomposition butterfly factorization. We will describe IDBF in detail in this section. For the sake of simplicity, we assume that $N = 2^L n_0$, ...
where $L$ is an even integer, and $n_0 = O(1)$ is the number of column or row indices in a leaf in the dyadic trees of row and column spaces, i.e., $T_X$ and $T_Ω$, respectively. Let us briefly introduce the main ideas of designing $O(\frac{k^3}{n_0} N \log N)$ IDBF using a linear ID. In IDBF, we compute $O(\log N)$ levels of low-rank submatrix factorizations. At each level, according to the matrix partition by the dyadic trees in column and row (see Figure 1 for an example), there are $\frac{N}{n_0}$ low-rank submatrices. Linear IDs only require $O(k^3)$ operations for each submatrix, and hence at most $O(\frac{k^3}{n_0} N)$ for each level of factorization, and $O(\frac{k^3}{n_0} N \log N)$ for the whole IDBF. There are two differences between IDBF and other BFs [12, 13, 14] as follows:

1. The order of factorization is from the leaf-root and root-leaf levels of matrix partitioning (e.g., the left and right panels of Figure 1) and moves towards the middle level of matrix partitioning (e.g., the middle panel of Figure 1).
2. Linear IDs are organized in an appropriate way such that it is cheap in terms of both memory and operations to provide all necessary information for each level of factorization.

In what follows, uppercase letters will generally denote matrices, while lowercase letters $c, p, q, r,$ and $s$ denote ordered sets of indices. For a given index set $c$, its cardinality is written $|c|$. Given a matrix $A$, the submatrix is $A_{pq}$, $A_{p,q}$, or $A(p,q)$, with rows and columns restricted to the index sets $p$ and $q$, respectively. We also use the notation $A_{s,t}$ to denote the submatrix with columns restricted to $s$. $s : t$ is an index set containing indices $\{s, s+1, s+2, \ldots, t-1, t\}$.

2.1. Linear scaling interpolative decompositions. Interpolative decomposition and other low-rank decomposition techniques [1, 3, 38] are important elements in modern scientific computing. These techniques usually require $O(kmn)$ arithmetic operations in order to obtain a rank $k = O(1)$ matrix factorization that approximates a matrix $A \in \mathbb{C}^{m \times n}$. Linear scaling randomized techniques can reduce the cost to $O(k(m + n))$ [39]. The paper [40] further shows that in the CUR low-rank approximation $A \approx CUR$, where $C = A_{:,c}$, $R = A_{r,:}$, and $U \in \mathbb{C}^{k \times k}$ with $|c| = |r| = k$, if only $U, c,$ and $r$ are needed, there exists an $O(k^3)$ algorithm for constructing $U, c,$ and $r$.

In the construction of IDBF, we use an $O(nk^3)$ linear scaling column ID to construct $V$, and we select skeleton indices $q$ such that $A \approx A_{:,q}V$ when $n \ll m$. Similarly, we can construct a row ID $A \approx UA_q$ in $O(mk^3)$ operations when $m \ll n$. As in [39, 40], randomized sampling can be applied to reduce the quadratic computational cost to linear. Here we present a simple lemma of ID to motivate the proposed linear scaling ID.

**Lemma 1.** For a matrix $A \in \mathbb{C}^{m \times n}$ with rank $k \leq \min\{m, n\}$, there exists a partition of the column indices of $A$, $p \cup q$ with $|q| = k$, and a matrix $T \in \mathbb{C}^{k \times (n-k)}$, such that $A_{:,p} = A_{:,q}T$.

**Proof.** A rank-revealing QR decomposition of $A$ gives

$$AA = QR = Q[R_1 \quad R_2],$$

where $Q \in \mathbb{C}^{m \times k}$ is an orthogonal matrix, $R \in \mathbb{C}^{k \times n}$ is upper triangular, and $A \in \mathbb{C}^{n \times n}$ is a carefully chosen permutation matrix such that $R_1 \in \mathbb{C}^{k \times k}$ is nonsingular. Let

$$A_{:,q} = QR_1,$$
and then let

\begin{equation}
A_{:,p} = QR_2 = QR_3 R_1^{-1} R_2 = A_{:,q} T,
\end{equation}

where

\begin{equation}
T = R_1^{-1} R_2.
\end{equation}

\(A_{:,p} = A_{:,q} T\) in Lemma 1 is equivalent to the traditional form of a column ID,

\begin{equation}
A = A_{:,q} [I \ T] A^* := A_{:,q} V,
\end{equation}

where * denotes the conjugate transpose of a matrix. We call \(p\) and \(q\) redundant and skeleton indices, respectively. \(V\) can be understood as a column interpolation matrix.

Our goal for linear scaling ID is to construct in \(O(k^2 n)\) operations and \(O(kn)\) memory the skeleton index set \(q\), the redundant index set \(p\), \(T\), and \(\Lambda\).

For a tall skinny matrix \(A\), i.e., \(m \gg n\), the rank-revealing QR decomposition of \(A\) in (2) typically requires \(O(kmn)\) operations. To reduce the complexity to \(O(k^2 n)\), we actually apply the rank-revealing QR decomposition to \(A_s\):

\begin{equation}
A_{s:,\Lambda} = QR = Q[R_1 \ R_2],
\end{equation}

where \(s\) is an index set containing \(tk\) carefully selected rows of \(A\), where \(t\) is an oversampling parameter. In many applications of interest, these rows can be chosen independently and uniformly from the row space, as in the sublinear CUR in [40] or the linear scaling algorithm in [39], or they can be chosen from the mock-Chebyshev grids (the subset of points taken from a given equispaced set that are nearest neighbors of actual Chebyshev nodes) of the row indices as in [17, 41, 42]. In fact, numerical results show that mock-Chebyshev points lead to a more efficient and accurate ID than randomly sampled points when matrices are from physical systems in a mesh domain. After the rank-revealing QR decomposition, the other steps to generate \(T\) and \(\Lambda\) take only \(O(k^2 n)\) operations since \(R_1\) in (5) is an upper triangular matrix.

In practice, the true rank of \(A\) is not available, i.e., \(k\) is unknown. In this case, the above computation procedure should be applied with some test rank \(k \leq n\). Furthermore, we are often interested in an ID with a numerical rank \(k_\epsilon\) specified by an accuracy parameter \(\epsilon\), i.e.,

\begin{equation}
\| A - A_{:,q} V \|_2 \leq O(\epsilon)
\end{equation}

with \(T \in \mathbb{C}^{k_\epsilon \times (n-k_\epsilon)}\) and \(V \in \mathbb{C}^{k_\epsilon \times n}\). We can choose

\begin{equation}
k_\epsilon = \min\{k : R_1(k,k) \leq \epsilon R_1(1,1)\},
\end{equation}

where \(R_1\) is given by the rank-revealing QR factorization in (7). Then define

\begin{equation}
T = (R_1(1 : k_\epsilon, 1 : k_\epsilon))^{-1} [R_1(1 : k_\epsilon, k_\epsilon + 1 : k) \ R_2(1 : k_\epsilon,:)] \in \mathbb{C}^{k_\epsilon \times (n-k_\epsilon)}
\end{equation}

and

\begin{equation}
V = [I \ T] \Lambda^* \in \mathbb{C}^{k_\epsilon \times n}.
\end{equation}

Correspondingly, let \(q\) be the index set such that

\begin{equation}
A_{:,q} = QR_1(1 : k_\epsilon, 1 : k_\epsilon),
\end{equation}

\(\epsilon\) and \(\xi\) are positive numbers that represent the accuracy and the number of significant figures, respectively.
and let $p$ be the complementary set of $q$; then $q$ and $V$ satisfy the requirement in (8). We refer to this linear scaling column ID with an accuracy tolerance $\epsilon$ and a rank parameter $k$ as $(\epsilon, k)$-cID. For convenience, we will drop the term $(\epsilon, k)$ when it is not necessary to specify it.

For a short and fat matrix $A \in \mathbb{C}^{m \times n}$ with $m \ll n$, a similar row ID

$$A \approx \Lambda[I T]^r A_q := U A_q,$$

can be devised similarly with $O(k^2 m)$ operations and $O(k m)$ memory. We refer to this linear scaling row ID as $\epsilon$-rID and to $U$ as the row interpolation matrix.

### 2.2. Leaf-root complementary skeletonization.

For a complementary low-rank matrix $A$, we introduce the leaf-root complementary skeletonization (LRCS)

$$A \approx U S V$$

via cIDs of the submatrices corresponding to the leaf-root levels of the column-row dyadic trees (e.g., see the associated matrix partition in Figure 2 (right)) and rIDs of the submatrices corresponding to the root-leaf levels of the column-row dyadic trees (e.g., see the associated matrix partition in Figure 2 (middle)). We always assume that IDs in this section are applied with a rank parameter $k = O(1)$. We will not specify $k$ again in the following discussion.

Suppose that at the leaf level of the row (and column) dyadic trees, the row index set $r$ (and the column index set $c$) of $A$ are partitioned into leaves $\{r_i\}_{1 \leq i \leq m}$ (and $\{c_i\}_{1 \leq i \leq m}$) as follows:

$$\begin{align*}
    r &= [r_1, r_2, \ldots, r_m] \quad (\text{and } c = [c_1, c_2, \ldots, c_m]),
\end{align*}$$

with $|r_i| = n_0$ (and $|c_i| = n_0$) for all $1 \leq i \leq m$, where $m = 2^L = \frac{N}{n_0}$, $L = \log_2 N - \log_2 n_0$, and $L + 1$ is the depth of the dyadic trees $T_X$ (and $T_\Omega$). Figure 2 shows examples of row and column dyadic trees with $m = 16$. We apply rID to each $A_{r_i}$ to obtain the row interpolation matrix in its ID and denote it as $U_i$; the associated skeleton indices of the ID are denoted as $\hat{r}_i \subset r_i$. Let

$$\hat{r} = [\hat{r}_1, \hat{r}_2, \ldots, \hat{r}_m];$$

then $A_{\hat{r}_i}$ is the important skeleton of $A$, and we have

$$A \approx \begin{pmatrix}
    U_1 \\ U_2 \\ \vdots \\ U_m
\end{pmatrix}
\begin{pmatrix}
    A_{\hat{r}_1, c_1} & A_{\hat{r}_1, c_2} & \cdots & A_{\hat{r}_1, c_m} \\
    A_{\hat{r}_2, c_1} & A_{\hat{r}_2, c_2} & \cdots & A_{\hat{r}_2, c_m} \\
    \vdots & \vdots & \ddots & \vdots \\
    A_{\hat{r}_m, c_1} & A_{\hat{r}_m, c_2} & \cdots & A_{\hat{r}_m, c_m}
\end{pmatrix}
:= UM.$$

Similarly, cID is applied to each $A_{\hat{r}_i, c_j}$ to obtain the column interpolation matrix $V_j$ and the skeleton indices $\hat{c}_j \subset c_j$ in its ID. Then, finally, we form the LRCS of $A$ as

$$A \approx \begin{pmatrix}
    U_1 \\ U_2 \\ \vdots \\ U_m
\end{pmatrix}
\begin{pmatrix}
    A_{\hat{r}_1, \hat{c}_1} & A_{\hat{r}_1, \hat{c}_2} & \cdots & A_{\hat{r}_1, \hat{c}_m} \\
    A_{\hat{r}_2, \hat{c}_1} & A_{\hat{r}_2, \hat{c}_2} & \cdots & A_{\hat{r}_2, \hat{c}_m} \\
    \vdots & \vdots & \ddots & \vdots \\
    A_{\hat{r}_m, \hat{c}_1} & A_{\hat{r}_m, \hat{c}_2} & \cdots & A_{\hat{r}_m, \hat{c}_m}
\end{pmatrix}
\begin{pmatrix}
    V_1 \\ V_2 \\ \vdots \\ V_m
\end{pmatrix}
:= USV.
A dyadic tree with one depth less (see Figure 6 for an example of a new compressible version of the row (and column) dyadic trees is again a complementary matrix. The row (and column) dyadic trees are generated and stored via the skeleton of row and column index sets. Hence, it only takes \( O(\frac{n^2}{m^0}) \) memory to generate and store the factorization in (14), since there are \( 2m = \frac{2N}{m^0} \) IDs in total.

For a concrete example, Figure 3 visualizes the nonzero pattern of the LRCS in (14) of the complementary low-rank matrix \( A \) in Figure 2.

The novelty of the LRCS is that \( M \) and \( S \) are not computed explicitly; instead, they are generated and stored via the skeleton of row and column index sets. Hence, it only takes \( O\left(\frac{n^2}{m^0}N\right) \) operations and \( O\left(\frac{n^2}{m^0}N\right) \) memory to generate and store the factorization in (14), since there are \( 2m = \frac{2N}{m^0} \) IDs in total.

It is worth emphasizing that in the LRCS of a complementary matrix \( A \approx USV \), the matrix \( S \) is again a complementary matrix. The row (and column) dyadic trees \( T_X \) (and \( T_{\Omega} \)) of \( S \) is the compressed version of the row (and column) dyadic trees \( T_X \) (and \( T_{\Omega} \)) of \( A \). Figure 4 (see also Figure 5) visualizes the relation of \( T_X \) and \( T_X \) (or \( T_{\Omega} \) and \( T_{\Omega} \)) for the complementary matrix \( A \) in Figure 2. \( T_X \) (or \( T_{\Omega} \)) is not compressible at the leaf level of \( T_X \) (or \( T_{\Omega} \)), but it is compressible if it is considered as a dyadic tree with one depth less (see Figure 6 for an example of a new compressible dyadic tree with one depth less).

### 2.3. Matrix splitting with complementary skeletonization.

Here we describe another elementary idea of IDBF that is applied repeatedly: matrix splitting with complementary skeletonization (MSCS). A complementary low-rank matrix \( A \) (with row and column dyadic trees \( T_X \) and \( T_{\Omega} \) of depth \( L \) and with \( m = 2^L \) leaves) can be split into a \( 2 \times 2 \) block matrix

\[
A = \begin{pmatrix}
A_{11} & A_{12} \\
A_{21} & A_{22}
\end{pmatrix}
\]

according to the nodes of the second level of the dyadic trees \( T_X \) and \( T_{\Omega} \) (those nodes right next to the root level). By the complementary low-rank property of \( A \), we know...
Fig. 4. Left: The dyadic tree $T_X$ of the row space with leaves $\{r_i\}_{1 \leq i \leq 16}$ denoted as in (12) for the example in Figure 2. Right: Selected important rows $\{\bar{r}_i\}_{1 \leq i \leq 16}$ in (13) are marked in red (level 5) and can be traced in previous levels (also marked in red in levels 1–4). Those important rows of $T_X$ naturally form a compressed dyadic tree shown in red. (See online version for color.)

Fig. 5. Left: The dyadic tree $T_\Omega$ of the column space with leaves $\{c_i\}_{1 \leq i \leq 16}$ denoted as in (12) for the example in Figure 2. Right: Selected important rows $\{\bar{c}_i\}_{1 \leq i \leq 16}$ are marked in red (level 5) and can be traced in previous levels (also marked in red in levels 1–4). Those important columns of $T_\Omega$ naturally form a compressed dyadic tree shown in red. (See online version for color.)

Fig. 6. Left: The compressed dyadic tree of $T_X$ of the row space in Figure 4. Level 5 is not compressible. Middle left: Combining adjacent leaves at level 5, i.e., $\bar{r}_i = \bar{r}_{2i-1} \cup \bar{r}_{2i}$, forms a compressible dyadic tree with depth 4. Middle right: The compressed dyadic tree of $T_\Omega$ of the column space in Figure 5. Level 5 is not compressible. Right: Combining adjacent leaves at level 5, i.e., $\bar{c}_i = \bar{c}_{2i-1} \cup \bar{c}_{2i}$, forms a compressible dyadic tree with depth 4.

that $A_{ij}$ is also complementary low-rank, for all $i$ and $j$, with row and column dyadic trees $T_{X,ij}$ and $T_{\Omega,ij}$ of depth $L - 1$ and with $m/2$ leaves.

Suppose $A_{ij} \approx U_{ij}S_{ij}V_{ij}$, for $i,j = 1,2$, is the LRCS of $A_{ij}$. Then $A$ can be
factorized as $A \approx USV$, where

$$
U = \begin{pmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{pmatrix},
$$

$$
S = \begin{pmatrix} S_{11} & S_{21} \\ S_{12} & S_{22} \end{pmatrix},
$$

$$
V = \begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix}.
$$

The factorization in (16) is referred to as the MSCS in this paper. Recall that the middle factor $S$ is not explicitly computed, resulting in a linear scaling algorithm for forming (16). Figure 7 visualizes the MSCS of the complementary low-rank matrix $A$ with dyadic trees of depth 5 and 16 leaf nodes shown in Figure 2.

2.4. Recursive matrix splitting with complementary skeletonization.

Now we apply MSCS recursively to get the full IDBF of a complementary low-rank matrix $A$ (with row and column dyadic trees $T_X$ and $T_Y$ of depth $L$ and with $m = 2^L$ leaves). As in (16), suppose we have constructed the first level of MSCS, and denote it as

$$
A \approx U^L S^L V^L
$$

with

$$
U^L = \begin{pmatrix} U_{11}^L & U_{12}^L \\ U_{21}^L & U_{22}^L \end{pmatrix},
$$

$$
S^L = \begin{pmatrix} S_{11}^L & S_{21}^L \\ S_{12}^L & S_{22}^L \end{pmatrix},
$$

$$
V^L = \begin{pmatrix} V_{11}^L & V_{12}^L \\ V_{21}^L & V_{22}^L \end{pmatrix},
$$

as in (16).
Suppose that at the leaf level of the row and column dyadic trees, the row index set $r$ and the column index set $c$ of $A$ are partitioned into leaves $\{r_i\}_{1 \leq i \leq m}$ and $\{c_i\}_{1 \leq i \leq m}$ as in (12). By the rIDs and cIDs applied in the construction of (17), we have obtained skeleton index sets $\hat{r}_i \subset r_i$ and $\hat{c}_i \subset c_i$. Then

$$
S^L_{ij} = \begin{pmatrix}
A_{\hat{r}_{(i-1)m/2+1}\hat{c}_{(j-1)m/2+1}} & \cdots & A_{\hat{r}_{(i-1)m/2+1}\hat{c}_{jm/2}} \\
\vdots & \ddots & \vdots \\
A_{\hat{r}_{im/2}\hat{c}_{(j-1)m/2+1}} & \cdots & A_{\hat{r}_{im/2}\hat{c}_{jm/2}}
\end{pmatrix}
$$

for $i, j = 1, 2$.

As explained in section 2.2, each nonzero block $S^L_{ij}$ in $S^L$ is a submatrix of $A_{ij}$ consisting of important rows and columns of $A_{ij}$ for $i, j = 1, 2$. Hence, $S^L_{ij}$ inherits the complementary low-rank property of $A_{ij}$ and is itself a complementary low-rank matrix. Suppose $T_X,ij$ and $T_{\Omega,ij}$ are the dyadic trees of the row and column spaces of $A_{ij}$ with $m/2$ leaves and $L - 1$ depth; then according to section 2.2, $S^L_{ij}$ has compressible row and column dyadic trees $T_X,ij$ and $T_{\Omega,ij}$ with $m/4$ leaves and $L - 2$ depth.

Next, we apply MSCS to each $S^L_{ij}$ in a recursive way. In particular, we divide each $S^L_{ij}$ into a $2 \times 2$ block matrix according to the nodes at the second level of its row and column dyadic trees:

$$
S^L_{ij} = \begin{pmatrix}
(S^L_{ij})_{11} & (S^L_{ij})_{12} \\
(S^L_{ij})_{21} & (S^L_{ij})_{22}
\end{pmatrix}
$$

After constructing the LRCS of the $(k, \ell)$th block of $S^L_{ij}$, i.e.,

$$(S^L_{ij})_{k\ell} \approx (U^L_{ij}S^L_{ij}V^L_{ij})_{k\ell}$$

we assemble matrices to obtain the MSCS of $S^L_{ij}$ as follows:

$$
S^L_{ij} \approx U^L_{ij}^{-1}S^L_{ij}^{-1}V^L_{ij}^{-1},
$$

where

$$
U^L_{ij}^{-1} = \begin{pmatrix}
(U^L_{ij})_{11} & (U^L_{ij})_{12} \\
(U^L_{ij})_{21} & (U^L_{ij})_{22}
\end{pmatrix},
$$

$$
S^L_{ij}^{-1} = \begin{pmatrix}
(S^L_{ij})_{11} & (S^L_{ij})_{12} \\
(S^L_{ij})_{21} & (S^L_{ij})_{22}
\end{pmatrix},
$$

$$
V^L_{ij}^{-1} = \begin{pmatrix}
(V^L_{ij})_{11} & (V^L_{ij})_{12} \\
(V^L_{ij})_{21} & (V^L_{ij})_{22}
\end{pmatrix},
$$

according to section 2.3.

Finally, we organize the factorizations in (21) for all $i, j = 1, 2$ to form a factorization of $S^L$ as

$$
S^L \approx U^L_{ij}^{-1}S^L_{ij}^{-1}V^L_{ij}^{-1},
$$
Fig. 8. The visualization of the recursive MSCS of $S^L = U^{L-1} S^{L-1} V^{L-1}$ in (23) when $A$ is a complementary low-rank matrix with dyadic trees of depth 5 and 16 leaf nodes as shown in Figure 2.

where

$$
U^{L-1} = \begin{pmatrix}
U_{11}^{L-1} & U_{12}^{L-1} \\
U_{21}^{L-1} & U_{11}^{L-1}
\end{pmatrix},
$$

$$
S^{L-1} = \begin{pmatrix}
S_{11}^{L-1} & S_{12}^{L-1} \\
S_{21}^{L-1} & S_{22}^{L-1}
\end{pmatrix},
$$

$$
V^{L-1} = \begin{pmatrix}
V_{11}^{L-1} & V_{12}^{L-1} \\
V_{21}^{L-1} & V_{22}^{L-1}
\end{pmatrix},
$$

leading to a second level factorization of $A$:

$$
A \approx U^L U^{L-1} S^{L-1} V^{L-1} V^L.
$$

Figure 8 visualizes the recursive MSCS of $S^L$ in (23) when $A$ is a complementary low-rank matrix with dyadic trees of depth 5 and 16 leaf nodes as shown in Figure 2.

Comparing (17), (18), (23), and (24), we can see a fractal structure in each level of the middle factor $S^L$ for $\ell = L$ and $L - 1$. For example, in (24) (see Figure 8 for its visualization), $S^{L-1}$ has four submatrices $S_{ij}^{L-1}$ with the same structure as $S^L$ for all $i$ and $j$. $S_{ij}^{L-1}$ can be factorized into a product of three matrices with the same sparsity structure as the factorization $S^L \approx U^{L-1} S^{L-1} V^{L-1}$. Hence, we can apply MSCS recursively to each $S^L$ and assemble matrix factors hierarchically for $\ell = L, L - 1, \ldots, L/2$, to obtain

$$
A \approx U^L U^{L-1} \cdots U^h S^h V^h \cdots V^{L-1} V^L,
$$

where $h = L/2$. In the $h$th recursive MSCS, $S^h$ has $2^{2(L-\ell+1)}$ dense submatrices with compressible row and column dyadic trees with $m \frac{m}{2^{2(L-\ell+1)}}$ leaves and depth $L - 2(L - \ell + 1)$. Hence, the recursive MSCS stops after $h = L/2$ iterations when $S^h$ no longer contains any compressible submatrix.

When $S^\ell$ is still compressible, since there are $2^{2(L-\ell+1)}$ dense submatrices and each contains $m \frac{m}{2^{2(L-\ell+1)}}$ leaves, there are $2^{2(L-\ell+1)} m \frac{m}{2^{2(L-\ell+1)}} m = N \frac{m}{n_0}$ low-rank submatrices to be factorized. Linear IDs only require $O(k^3)$ operations for each low-rank submatrix, and hence at most $O(\frac{k^3}{n_0} N)$ for each level of factorization, and $O(\frac{k^3}{n_0} N \log N)$ for the whole IDBF.
3. Numerical results. This section presents several numerical examples to demonstrate the effectiveness of the algorithms proposed above. The first three examples are complementary low-rank matrices coming from nonuniform Fourier transforms, Fourier integral operators, and special function transforms. The last two examples are hierarchical complementary matrices \cite{30} from 2D Helmholtz boundary integral methods in the high-frequency regime. All implementations were done are in MATLAB on a server computer with a single thread and 3.2 GHz CPU. This new framework will be incorporated into the ButterflyLab\footnote{Available at https://github.com/ButterflyLab.} in the future.

Let \( \{ u^d(x), x \in X \} \) and \( \{ u^a(x), x \in X \} \) denote the results given by the direct matrix-vector multiplication and the butterfly factorization. The accuracy of applying the butterfly factorization algorithm is estimated by the relative error

\[
\epsilon^a = \sqrt{\frac{\sum_{x \in S} |u^a(x) - u^d(x)|^2}{\sum_{x \in S} |u^d(x)|^2}},
\]

where \( S \) is a point set of size 256 randomly sampled from \( X \). In all of our examples, the oversampling parameter \( t \) in the linear scaling ID is set to 1, and the number of points in a leaf node is set to \( n_0 = 8 \). Then the number of randomly sampled grid points in the ID is equal to the rank parameter \( k \), which we will here also call the truncation rank.

\textsc{Example 1.} Our first example evaluates a 1D Fourier integral operator (FIO) of the form

\[
\begin{align*}
\Phi(x, \xi) &= x \cdot \xi + c(x) |\xi|, \\
c(x) &= (2 + 0.2 \sin(2\pi x))/16.
\end{align*}
\]

The discretization of (27) is

\[
u(x_i) = \sum_{\xi_j} e^{2\pi i \Phi(x_i, \xi_j)} \hat{f}(\xi_j), \quad i, j = 1, 2, \ldots, N,
\]

where \( \{ x_i \} \) and \( \{ \xi_j \} \) are uniformly distributed points in \([0, 1)\) and \([-N/2, N/2)\) following

\[
x_i = (i - 1)/N \quad \text{and} \quad \xi_j = j - 1 - N/2.
\]

Equation (29) can be represented in matrix form as \( u = Kg \), where \( u_i = u(x_i) \), \( K_{ij} = e^{2\pi i \Phi(x_i, \xi_j)} \), and \( g_j = \hat{f}(\xi_j) \). The matrix \( K \) satisfies the complementary low-rank property with a rank parameter \( k \) independent of the problem size \( N \) when \( \xi \) is sufficiently far away from the origin, as proved in \cite{35, 43}. To make the presentation simpler, we will directly apply IDBF to the whole \( K \) instead of performing a polar transform as in \cite{35} or applying IDBF hierarchically as in \cite{43}. Hence, due to the nonsmoothness of the \( \Phi(x, \xi) \) at \( \xi = 0 \), submatrices intersecting with or close to the line \( \xi = 0 \) have a local rank increasing slightly in \( N \), while other submatrices have rank independent of \( N \).
Figures 9–11 summarize the results of this example for different grid sizes $N$. To compare IDs with mock-Chebyshev points and randomly selected points in different cases, Figure 9 shows the results for tolerance $\epsilon = 10^{-6}$ in (9) and when the truncation rank $k$ is the smallest size of a submatrix (i.e., $k = \min\{m, n\}$ for a submatrix of size $m \times n$); Figure 10 shows the results for $\epsilon = 10^{-6}$ and $k = 30$; Figure 11 shows the results for $\epsilon = 10^{-15}$ and $k = 30$. Note that the accuracy of IDBF is expected to be $O(\epsilon)$, which may not be guaranteed, since the overall accuracy of IDBF is determined by all IDs in a hierarchical manner. Furthermore, if the rank parameter $k$ is too small for some low-rank matrices, then the error of the corresponding ID will propagate through the whole IDBF process and increase the error of the IDBF.

We see that the IDBF applied to the whole matrix $K$ has $O(N \log^2(N))$ factorization and application time in all cases with different parameters. The running time agrees with the scaling of the number of nonzero entries required in the data-sparse representation. In fact, when $N$ is large enough, the number of nonzero entries in the IDBF tends to scale as $O(N \log N)$, which means that the numerical scaling can approach $O(N \log N)$ in both factorization and application when $N$ is large enough. IDBF via IDs with mock-Chebyshev points is much more accurate than IDBF via IDs with random samples. The running times for three kinds of parameter pairs $(\epsilon, k)$ are almost the same. For the purpose of numerical accuracy, we prefer IDs with mock-Chebyshev points with $(\epsilon, k) = (10^{-15}, 30)$. Hence, in later examples we will only present numerical results for IDs with mock-Chebyshev points.

**Example 2.** Next, we provide an example of a special function transform, the evaluation of Schlömilch expansions [44] at $g_k = \frac{k-1}{N}$ for $1 \leq k \leq N$:

$$
(31) \quad u_k = \sum_{n=1}^{N} c_n J_\nu(g_k \omega_n),
$$

where $J_\nu$ is the Bessel function of the first kind with parameter $\nu = 0$, and $\omega_n = n\pi$. It is demonstrated in [13] that (31) can be represented via a matvec $u = Kg$, where
\(K\) satisfies the complementary low-rank property. An arbitrary entry of \(K\) can be calculated in \(O(1)\) operations [45], and hence IDBF is suitable for accelerating the matvec \(u = Kg\). Other, similar examples when \(\nu \neq 0\) can be found in [44], and they can also be evaluated by IDBF with the same operation counts.

Figure 12 summarizes the results of this example for different problem sizes \(N\) with different parameter pairs \((\epsilon, k)\). The results show that IDBF applied to this example has \(O(N\log^2(N))\) factorization and application time. The running time agrees with the scaling of the number of nonzero entries required in the data-sparse representation to guarantee the approximation accuracy. In fact, when \(N\) is large enough, the number
of nonzero entries in the IDBF tends to scale as $O(N \log N)$, which means that the numerical scaling can approach $O(N \log N)$ in both factorization and application when $N$ is large enough.

**Example 3.** In this example, we consider the 1D nonuniform Fourier transform as follows:

$$u_k = \sum_{n=1}^{N} e^{-2\pi i x_n \omega_k} g_n,$$

for $1 \leq k \leq N$, where $x_n$ is randomly selected in $[0, 1)$, and $\omega_k$ is randomly selected in $[-\frac{N}{2}, \frac{N}{2})$ according to uniform distributions in these intervals.

Figure 13 summarizes the results of this example for different grid sizes $N$ with different parameter pairs $(\epsilon, k)$. Numerical results show that IDBF admits at most $O(N \log^2(N))$ factorization and application time for the nonuniform Fourier transform. The running time agrees with the scaling of the number of nonzero entries required in the data-sparse representation. In fact, when $N$ is large enough, the number of nonzero entries in the IDBF tends to scale as $O(N \log N)$, which means that the numerical scaling can approach $O(N \log N)$ in both factorization and application when $N$ is large enough.

**Example 4.** The fourth example is from the electric field integral equation (EFIE) for analyzing scattering from a two-dimensional curve. Using the method of moments...
on a linear segmentation of the curve, the EFIE takes the form [12]

\[ Z x = b, \]

where \( Z \) is an impedance matrix with (up to scaling)

\[ Z_{ij} = \begin{cases} w_i w_j H_0^{(2)}(\kappa |\rho_i - \rho_j|) & \text{if } i \neq j, \\ w_i^2 \left[ 1 - \frac{i^2}{\pi} \ln \left( \frac{2w_i \kappa}{4\pi} \right) \right] & \text{if } i = j, \end{cases} \]

where \( \epsilon \approx 2.718, \gamma \approx 1.781 \) is the exponential of the Euler–Mascheroni constant, \( \kappa = 2\pi/\lambda_0 \) is the wavenumber, \( \lambda_0 \) represents the free-space wavelength, \( H_0^{(2)} \) denotes the zeroth-order Hankel function of the second kind, \( w_i \) is the length of the \( i \)th linear segment of the scatterer object, and \( \rho_i \) is the center of the \( i \)th segment.

It was shown in [12, 30] that \( Z \) admits a HODLR-type complementary low-rank property; i.e., off-diagonal blocks are complementary low-rank matrices. As we mentioned in section 1, [12] compresses and applies the impedance matrix within \( O(N \log^2 N) \) operations. Based on [12], the authors of [30] developed an \( O(N^{1.5} \log N) \) direct solver, which leverages a randomized butterfly scheme to compress blocks corresponding to near- and far-field interactions, to invert the impedance matrix. The construction of the direct solver also involves fast matvec of complementary low-rank matrices. To show the potential application of IDBF in the direct solver, we use IDBF to compress and apply the impedance matrix.

---

Fig. 13. Numerical results for the NUFFT given in (32). \( N \) is the size of the matrix, \( \text{nnz} \) is the number of nonzero entries in the butterfly factorization, and \( \text{err} \) is the approximation error of the IDBF matvec. Top row: \( (\epsilon, k) = (10^{-6}, \min\{m,n\}) \). Middle row: \( (\epsilon, k) = (10^{-6}, 30) \). Bottom row: \( (\epsilon, k) = (10^{-15}, 30) \).
Figure 14. The two scatterers used in Examples 4 and 5. (a) A spiral object. (b) A round object with a hole in center which is the port.

Figure 15. Numerical results for the 2D EFIE. $N$ is the size of the matrix, $\text{nnz}$ is the number of nonzero entries in the butterfly factorization, and $\text{err}$ is the approximation error of the matvec by hierarchically applying IDBF.

Figure 15 shows the results of the fast matvec of the impedance matrix from a 2D EFIE generated with a spiral object, as shown in Figure 14(a). We vary the number of segments $N$ and let $\kappa = O(N)$ in the construction of $Z$. In the IDBF, we use truncation rank $k = 10$ and tolerance $\epsilon = 10^{-4}$ in IDs with mock-Chebyshev points. Numerical results verify the $O(N \log^2(N))$ scaling for both the factorization and application of the new HODLR-type butterfly factorization by IDBF.

Example 5. The fifth example is from the combined field integral equation (CFIE). Similarly to the ideas in [12, 30] for EFIE, we verify that the impedance matrix of the CFIE\(^2\) by the method of moments for analyzing scattering from 2D objects also admits a HODLR-type complementary low-rank property. Applying a HODLR-type butterfly factorization by IDBF, we obtain $O(N \log^2(N))$ scaling for both the factorization and application time for impedance matrices of CFIEs. This makes it possible to design efficient iterative solvers to solve the linear system for the impedance matrix. Figure 16 shows the results of the fast matvec of the impedance matrix from a 2D CFIE generated with a round object as shown in Figure 14(b). We vary grid sizes $N$ with truncation rank $k = 10$ and tolerance $\epsilon = 10^{-4}$ in IDs with mock-Chebyshev points. Numerical results verify the $O(N \log^2(N))$ scaling for both the factorization and application of the new HODLR-type butterfly factorization by IDBF.

4. Conclusion and discussion. This paper introduces IDBF as a data-sparse approximation of complementary low-rank matrices. It represents such an $N \times N$ dense matrix as a product of $O(\log N)$ sparse matrices. The factorization and application time and the memory of IDBF all scale as $O(N \log N)$. The order of factorization

\(^2\)Codes for generating the impedance matrix are from MATLAB package “emsolver” available at https://github.com/dsmi/emsolver.
is from the leaf-root and root-leaf levels of matrix partitioning (e.g., the left and right panels in Figure 1) and moves towards the middle level of matrix partitioning (e.g., the middle panel of Figure 1). Other orders of factorization are also possible, e.g., an order from the root of the column space to its leaves, an order from the root of the row space to its leaves, or an order from the middle level towards two sides. We leave the extensions of these $O(N \log N)$ IDBFs to the reader.

As shown by numerical examples, IDBF is able to construct the data-sparse representation of the HODLR-type complementary matrix in [30] in nearly linear scaling. These matrices arise widely in 2D high-frequency integral equation methods. By comparison of IDBF based on CUR and mock-Chebyshev points, we show that the IDBF with mock-Chebyshev points is more accurate and could be a good alternative to the factorization method in [30], since the factorization method in [30] shares the same spirit of IDBF based on CUR. IDBF could also improve the factorization accuracy of the hierarchical complementary matrix in [31] for 3D high-frequency boundary integral methods.

REFERENCES


Fig. 16. Numerical results for the 2D CFIE. $N$ is the size of the matrix, $nnz$ is the number of nonzero entries in the butterfly factorization, and $err$ is the approximation error of the matvec by hierarchically applying IDBF.


